Computing Feasible Vehicle Platooning Opportunities for Transport Assignments[⋆]

Sebastian van de Hoef[*] **Karl H. Johansson**[*]
Dimos V. Dimarogonas[*]

[*] *ACCESS Linnaeus Center and the School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden (e-mail:* {shvdh, kallej, dimos}@kth.se*).*

Abstract: Vehicle platooning facilitates the partial automation of vehicles and can significantly reduce fuel consumption. Mobile communication infrastructure makes it possible to dynamically coordinate the formation of platoons en route. We consider a centralized system that provides trucks with routes and speed profiles allowing them to dynamically form platoons during their journeys. For this to work, all possible pairs of vehicles that can platoon based on their location, destination, and other constraints have to be identified. The presented approach scales well to large vehicle fleets and realistic road networks by extracting features from the transport assignments of the vehicles and rules out a majority of possible pairs based on these features only. Merely a small number of remaining pairs are considered in depth by a complete and computationally expensive algorithm. This algorithm conclusively decides if platooning is possible for a pair based on the complete data associated with the two vehicles. We derive appropriate features for the problem and demonstrate the effectiveness of the approach in a simulation example.

Keywords: Transportation, Agents, Platooning, Spatial Networks

1. INTRODUCTION

Platooning is one of the fundamental building blocks for controlling connected vehicles. Vehicles are arranged in a convoy and the longitudinal spacing in the convoy is maintained with the help of automatic control. This simplifies the automation of the trailing vehicles and reduces their fuel consumption due to the slipstream effect (Bonnet and Fritz (2000)). While the low-level platoon control is well developed (Horowitz and Varaiya (2000); Hao and Barooah (2013)), the dynamic formation of platoons has only recently attracted the interest of researchers (Meisen et al. (2008); Hall and Chin (2005); Larson et al. (2015); Larsson et al. (2015); van de Hoef et al. (2015b,a)).

We envision an integrated system that centrally coordinates the formation of platoons. Trucks would connect to such a system via vehicle-to-infrastructure communication. The system continuously provides updated fuel-efficient routes and speed profiles to the connected trucks. These routes and speed profiles allow vehicles to meet on their journeys in order to form platoons. The computation of these routes and speed profiles takes various constraints such as arrival deadlines, speed-limits, and rest periods into account.

The contribution of this paper is a method to efficiently rule out the majority of transport assignment pairs that cannot form a platoon due to their geographic or temporal separation. The elements of the significantly smaller set of candidate pairs is then treated one by one.

The computation of routes and coordinated speed profiles happens in several stages. The first stage is the route calculation that produces a route to the destination for each transport assignment based on the current position of the vehicle or the planned start point of the transport assignment in the future. It also computes a set of possible trajectories along that route. The next stage identifies for which pairs of transport assignments the associated vehicles can meet in the future in order to platoon based on the output of the route calculation. This is the input to an optimization routine that determines the platoons formed and the corresponding trajectories. These trajectories are sent to the individual vehicles which execute them. This process is frequently repeated in order to account for new or changed assignments, vehicles deviating from the planned trajectories, and updated traffic information. Comparing all pairs individually in order to find out which pairs can platoon is only feasible for a small number of transport assignments. On the other hand, the number of transport assignments increases when larger geographic regions or longer time horizons are considered in the planning process. In 2011 over 1.7 million heavy trucks were in use (Asociación Española de Fabricantes de Automóviles y Camiones (2011)) in the European Union, a number so large that even a fraction of these can be challenging to coordinate.

A related problem to the one considered in this paper is to compute the collision of a large number of geometric objects, which has been considered in the field of computer

[⋆] This work was supported by the COMPANION EU project, the Knut and Alice Wallenberg Foundation, and the Swedish Research Council.

graphics (Jiménez et al. (2000); Bentley and Ottmann (1979); Edelsbrunner and Maurer (1981); Cohen et al. (1995); Liu et al. (2010)) and to some extent in the area of interest management for distributed virtual environments (Liu and Theodoropoulos (2014)). The way this problem is tackled might also be relevant for other spatial, large-scale multi-agent systems where the possible interactions need to be identified in real time and where the number of actual interactions is small compared to the number of agent pairs. Examples of such systems are collision avoidance of (autonomous) mobile agents (Roy and Tomlin (2006)) and ride-sharing systems (Agatz et al. (2012)). Similar to the application considered in this paper, these are large-scale spatially-distributed systems (Barthélemy (2011)) that are enabled by the rapid development of the communication infrastructure.

This paper is organized as follows. After having introduced the problem and related notation (Section 2), we first abstract approaches that have been developed in computer graphics (Section 3). We introduce the concept of features and classifiers that can indicate a pair of vehicles not being able to platoon on their routes. In Section 4 we derive a family of features for the problem setting introduced in Section 2. We demonstrate the method with a simulation study in Section 5.

2. PROBLEM FORMULATION

We proceed with introducing the problem setup considered in this paper. We model the road network as a directed graph $\mathcal{G}_\mathrm{r} = (\mathcal{N}_\mathrm{r}, \mathcal{E}_\mathrm{r})$ with nodes $\mathcal{N}_\mathrm{r}$ and edges $\mathcal{E}_\mathrm{r} \in \mathcal{N}_\mathrm{r} \times \mathcal{N}_\mathrm{r}$. Nodes correspond to intersections or endpoints in the road network and links correspond to road segments connecting these intersections. Each node in $\mathcal{N}_\mathrm{r}$ can be associated with a 2-D coordinate $\mathbf{P} : \mathcal{N}_\mathrm{r} \to \mathbb{R}^2$. Furthermore, we assume that the length of a road segment modeled by an edge equals the euclidean distance between the positions of the two nodes that comprise the edge.

We have K transport assignments and let $\mathcal{N}_\mathrm{c} = \{1, \ldots, K\}$ be an index set of all assignments. Each transport assignment consists of a start node $n^\mathrm{S} \in \mathcal{N}_\mathrm{r}$ and a destination node $n^\mathrm{D} \in \mathcal{N}_\mathrm{r}$. Furthermore, for each transport assignment, there is an earliest start time t^S and a latest arrival time t^D.

The input to the vehicles are paths and speed profiles, one for each vehicle, that *implement* the transport assignments. To that end, we define a trajectory (route and speed-profile) of a vehicle and the requirements for a trajectory to implement a transport assignment.

Definition 1. (Trajectory). A trajectory is a pair $(\mathbf{n}, \mathbf{t})$. The route $\mathbf{n} = n[1], \ldots, n[N]$ is a sequence of nodes in $\mathcal{N}_\mathrm{r}$ that describe a path in $\mathcal{G}_\mathrm{r}$. We refer to $\mathbf{n}$ as a route. The second element $\mathbf{t} = t[1], \ldots, t[N]$ is a sequence of time instances such that $t[a+1] - t[a] \geq \|\mathbf{P}(n[a+1]) - \mathbf{P}(n[a])\|_2 / v_\mathrm{max}$ for $a = 1, \ldots, N-1$, where v_max is the maximum speed. The number of nodes N may be different for different trajectories.

We neglect that the speed on a link is restricted by the speed on the adjacent links and that the maximum speed v_max depends on the link and on time. This is mainly for the ease of presentation but can as well be a reasonable simplification in the culling phase and can be accounted for in the routine that calculates the actual speed profiles. A trajectory $(\mathbf{n}, \mathbf{t})$ implements a transport assignment $(n^\mathrm{S}, n^\mathrm{D}, t^\mathrm{S}, t^\mathrm{D})$ if it starts after the start time at the start node of the transport assignment and arrives before the deadline at destination node of the transport assignment, i.e., if $n[1] = n^\mathrm{S}$, $n[N] = n^\mathrm{D}$, $t[1] \geq t^\mathrm{S}$, $t[N] \leq t^\mathrm{D}$. We assume that the route $\mathbf{n}$ of the trajectory is given, typically the shortest path. The coordination is restricted to adapting the speed profile, i.e., the sequence $\mathbf{t}$. We need to be able to test whether there are trajectories that platoon, i.e., partially coincide. To this end we calculate for each node in $\mathbf{n}$ a time interval. The element in $\mathbf{t}$ that corresponds to the node in $\mathbf{n}$ lies in this interval if $\mathbf{t}$ belongs to a trajectory that implements the transport assignment. Only when the intervals for two transport assignments overlap at a common node, the possibility of implementing trajectories that platoon exists. We denote the sequence of lower bounds on the elements of $\mathbf{t}$ as $\underline{\mathbf{t}} = \underline{t}[1], \ldots, \underline{t}[N]$ and the upper bounds as $\bar{\mathbf{t}} = \bar{t}[1], \ldots, \bar{t}[N]$. They are computed for $a = 1, \ldots, N$ as

$$\underline{t}[a] = t^\mathrm{S} + \sum_{m=1}^{a-1} \frac{\|\mathbf{P}(n[m+1]) - \mathbf{P}(n[m])\|_2}{v_\mathrm{max}} \qquad (1)$$

$$\bar{t}[a] = t^\mathrm{D} - \sum_{m=a}^{N-1} \frac{\|\mathbf{P}(n[m+1]) - \mathbf{P}(n[m])\|_2}{v_\mathrm{max}}. \qquad (2)$$

Next, we define a function that indicates whether platooning between two transport assignments is possible or not. This is the case if there is at least one common edge in the routes of the transport assignments where the time bounds of the two assignments overlap.

Definition 2. (Coordination Function). The coordination function $C : \mathcal{N}_\mathrm{c} \times \mathcal{N}_\mathrm{c} \to \{0, 1\}$ has the following properties. Let $\underline{\mathbf{t}}_i, \underline{\mathbf{t}}_j$ be lower bounds and $\bar{\mathbf{t}}_i, \bar{\mathbf{t}}_j$ be upper bounds on the node arrival times of transport assignments i and j according to (1), (2). Then it holds that $C(i,j) = 1$, if there are indices a, b such that $\mathbf{P}(n_i[a]) = \mathbf{P}(n_j[b])$ and $\mathbf{P}(n_i[a+1]) = \mathbf{P}(n_j[b+1])$, and $[\underline{t}_i[a], \bar{t}_i[a]] \cap [\underline{t}_j[b], \bar{t}_j[b]] \neq \emptyset$ and $[\underline{t}_i[a+1], \bar{t}_i[a+1]] \cap [\underline{t}_j[b+1], \bar{t}_j[b+1]] \neq \emptyset$. Otherwise $C(i,j) = 0$.

Comparing the routes and the time bounds in order to evaluate C, is straightforward but computationally expensive. We refer to this as the *exact algorithm.* The goal of this work is to find a scalable method for computing the set of all possible platoon pairs $\mathcal{C} = \{(i,j) \in \mathcal{N}_\mathrm{c} \times \mathcal{N}_\mathrm{c} : C(i,j) = 1\}$. Instead of iterating over all elements in $\mathcal{N}_\mathrm{c} \times \mathcal{N}_\mathrm{c}$ and using the exact algorithm, we propose to first efficiently compute an overapproximation $\hat{\mathcal{C}} \supset \mathcal{C}$ and then applying the exact algorithm.

3. CULLING

The key idea of our approach is to extract features from the routes and time bounds $(\mathbf{n}, \underline{\mathbf{t}}, \bar{\mathbf{t}})$ of the transport assignments to compute $\hat{\mathcal{C}}$. These features can be more efficiently processed than $\mathbf{n}, \underline{\mathbf{t}}, \bar{\mathbf{t}}$. Features are designed in a way that no platooning opportunity in $\mathcal{C}$ will be excluded from $\hat{\mathcal{C}}$, so that $\mathcal{C}$ can be computed from $\hat{\mathcal{C}}$ using the exact algorithm. However, there might be some additional elements in $\hat{\mathcal{C}}$

that do not actually correspond to platooning opportunities. We call these additional elements false-positives. The less false-positives there are in $\hat{\mathcal{C}}$, the faster is the computation of $\mathcal{C}$ from $\hat{\mathcal{C}}$. This approach is inspired by a related problem of detecting which pairs of a large number of geometric objects intersect or collide.

We consider two types of features. These are *interval features* and *binary features*. Interval features map each object to an interval. The corresponding classifier indicates an intersection between two objects if the intervals generated by the objects overlap. There are algorithms (Bentley and Ottmann (1979); Edelsbrunner and Maurer (1981)) that can compute this classifier for all object pairs more efficiently than checking each pair individually, if the number of reported intersecting pairs is small. Binary features map each object to a boolean value. The corresponding classifier indicates an intersection between two objects if the feature holds true for both objects. In Section 4, we derive appropriate features for the problem stated in Section 2.

The classifiers are aggregated using boolean connectives. We formalize this in the remainder of the section. Let $\mathcal{N}$ be a set of objects. We define a classifier as a function $c : \mathcal{N} \times \mathcal{N} \to \{0, 1\}$. If $c(i, j) = 0$, we call the combination of c and (i, j) a negative, and if $c(i, j) = 1$, we call it a positive. Let $g : \mathcal{N} \times \mathcal{N} \to \{0, 1\}$ be the ground truth which can be computed by the exact algorithm. If for a pair (i, j) we have $g(i, j) = 0$ and $c(i, j) = 1$, we call it a false-positive, and if $g(i, j) = 1$ and $c(i, j) = 0$, we call it a false-negative. Our aim is to design classifiers which yield no false negatives for all elements of $\mathcal{N} \times \mathcal{N}$ and few false-positives that have to be processed by the exact algorithm in addition to the true-positives.

We can identify two types of basic classifiers that are combined in a specific way in order to achieve the above objective. A classifier c is *required* if $\neg c(i, j) \Rightarrow \neg g(i, j)$ for all $i, j \in \mathcal{N} \times \mathcal{N}$. In some cases, we have to take into account a set of classifiers to conclude that g does not hold. A set of classifiers $\mathcal{D}$ is *required* if $\neg \bigvee_{c \in \mathcal{D}} c(i, j) \Rightarrow \neg g(i, j)$ for all $i, j \in \mathcal{N} \times \mathcal{N}$. It is straightforward to construct a required classifier from a required set of classifiers.

Proposition 1. If a set $\mathcal{D}$ of classifiers is required, then $\bigvee_{c \in \mathcal{D}} c$ is a required classifier.

We can combine two required classifiers into one required classifier that performs no worse than any of the required classifiers it is combined of.

Proposition 2. If c_1 and c_2 are required classifiers, then $c_{12} := c_1 \wedge c_2$ is a required classifier. Let $\bar{\mathcal{E}}_{12} = \{(i, j) \in \mathcal{N} \times \mathcal{N} : c_{12}(i, j) = 0\}$ be the set of negatives of c_{12} and let $\bar{\mathcal{E}}_1$, $\bar{\mathcal{E}}_2$ be the set of negatives for c_1 and c_2 respectively. Then $\bar{\mathcal{E}}_1 \subseteq \bar{\mathcal{E}}_{12}$ and $\bar{\mathcal{E}}_2 \subseteq \bar{\mathcal{E}}_{12}$.

Proof. For c_{12} to be required, we need to show that $\neg c_{12}(i, j) \Rightarrow \neg g(i, j)$ for all $i, j \in \mathcal{N} \times \mathcal{N}$. We have $(\neg c_1 \Rightarrow \neg g) \wedge (\neg c_2 \Rightarrow \neg g) = (c_1 \vee \neg c_1 \wedge \neg g) \wedge (c_2 \vee \neg c_2 \wedge \neg g) = c_1 \wedge c_2 \vee \neg g \wedge (\neg c_1 \wedge \neg c_2 \vee \neg c_1 \wedge c_2 \vee c_1 \wedge \neg c_2) = c_1 \wedge c_2 \vee \neg g \wedge (\neg c_1 \vee \neg c_2) = c_1 \wedge c_2 \vee \neg g \wedge \neg(c_1 \wedge c_2) = \neg(c_1 \wedge c_2) \Rightarrow \neg g = \neg c_{12} \Rightarrow \neg g$. Let $(i, j) \in \bar{\mathcal{E}}_1$. Then from the definition of $\bar{\mathcal{E}}_1$ we have that $c_1(i, j) = 0$. We have that $c_{12}(i, j) = c_1(i, j) \wedge c_2(i, j) = 0 \wedge c_2(i, j) = 0$. It follows from the definition of $\bar{\mathcal{E}}_{12}$ that $(i, j) \in \bar{\mathcal{E}}_{12}$. Similarly, we see that any element of $\bar{\mathcal{E}}_1$ is an element of $\bar{\mathcal{E}}_{12}$.

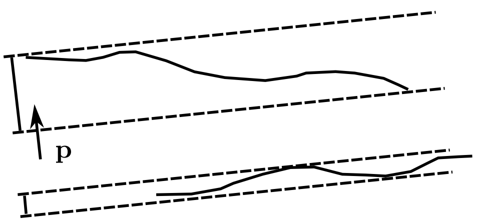

Fig. 1. Illustration of the projection feature. It shows how the two routes (solid lines) are projected onto a line in the direction of the vector **p**. The borders of the intervals are indicated with dashed lines. For illustration purposes the third dimension is omitted here.

In this manner, we can combine as many required classifiers as we want and have at our disposal. With each classifier we add, we potentially decrease the set of remaining candidates that need to be checked by the exact algorithm. There is a trade-off between doing more work to evaluate more classifiers and less instances which have to be processed by the exact algorithm (Liu et al. (2010)).

4. FEATURES AND CLASSIFIERS

In order to apply the results from Section 3, we need to specify appropriate features and classifiers based on these features for the problem stated in Section 2. Once we know how to compute appropriate features that yield required classifiers or required sets of classifiers, we can use the results from Section 3 to execute the culling phase. The remaining candidate pairs are passed on to the exact algorithm to compute $\mathcal{C}$. Hence, we will derive a selection of features and corresponding classifiers in this section. In Section 5, we will demonstrate these classifiers and combinations of them in a simulation example.

The first feature projects the possible trajectories on a line which yields an interval. Formally, we define this feature as follows.

Definition 3. Let $\mathbf{p} \in \mathbb{R}^3$ be a three dimensional vector which defines the orientation of the line which the trajectories are projected onto. Then the associated interval feature is defined as

$$\mathcal{I} = [\min_{\mathbf{v} \in \mathcal{R}} (\mathbf{p}^\mathsf{T} \mathbf{v}), \max_{\mathbf{v} \in \mathcal{R}} (\mathbf{p}^\mathsf{T} \mathbf{v})] \tag{3}$$

with

$$\mathcal{R} = \left\{ \begin{bmatrix} \mathbf{P}(n[1]) \\ \underline{t}[1] \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{P}(n[N]) \\ \underline{t}[N] \end{bmatrix}, \begin{bmatrix} \mathbf{P}(n[1]) \\ \bar{t}[1] \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{P}(n[N]) \\ \bar{t}[N] \end{bmatrix} \right\}. \tag{4}$$

This feature is illustrated in Figure 1. The projection vector **p** is a design choice. Proposition 2 allows us to combine arbitrarily many classifiers based on this kind of feature with different **p**.

Next, we establish that if for a pair of transport assignments the intervals do not overlap the coordination function is equal to zero. This allows us to define a required feature based on the overlap between these intervals.

Proposition 3. Let (i,j) refer to a pair of transport assignments. Let $\mathcal{I}_i, \mathcal{I}_j$ be the interval features according to (3) for the two transport assignments. Then $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset \Rightarrow C(i,j) = 0$.

Proof. According to Definition 2, $C(i,j) = 1$ implies that there must be indices a, b such that $\mathbf{P}(n_i[a]) = \mathbf{P}(n_j[b])$ and $[\underline{t}_i[a], \bar{t}_i[a]] \cap [\underline{t}_j[b], \bar{t}_j[b]] \neq \emptyset$, where $\mathbf{n}_i, \underline{\mathbf{t}}_i, \bar{\mathbf{t}}_i$ and $\mathbf{n}_j, \underline{\mathbf{t}}_j, \bar{\mathbf{t}}_j$ are the node sequences and time bounds of transport assignment i, j respectively. We have

$$[\underline{t}_i[a], \bar{t}_i[a]] \cap [\underline{t}_j[b], \bar{t}_j[b]] \neq \emptyset \Leftrightarrow \underline{t}_i[a] \leq \bar{t}_j[b] \wedge \underline{t}_j[b] \leq \bar{t}_i[a].$$

Let $\mathbf{p} = [p_1, p_2, p_3]^{\mathrm{T}}$, $\mathbf{P} = \mathbf{P}(n_i[a]) = \mathbf{P}(n_j[b])$, and $P^0 = [p_1, p_2]\mathbf{P}$. We have

$$\begin{aligned}
&\underline{t}_i[a] \leq \bar{t}_j[b] \wedge \underline{t}_j[b] \leq \bar{t}_i[a] \\
&\Rightarrow \min(p_3\underline{t}_i[a], p_3\bar{t}_i[a]) \leq \max(p_3\underline{t}_j[b], p_3\bar{t}_j[b]) \\
&\Rightarrow \min(p_3\underline{t}_i[a] + P^0, p_3\bar{t}_i[a] + P^0) \leq \max(p_3\underline{t}_j[b] + P^0, p_3\bar{t}_j[b] + P^0) \\
&\Rightarrow \min\left(\mathbf{p}^{\mathrm{T}}\begin{bmatrix}\mathbf{P}\\ \underline{t}_i[a]\end{bmatrix}, \mathbf{p}^{\mathrm{T}}\begin{bmatrix}\mathbf{P}\\ \bar{t}_i[a]\end{bmatrix}\right) \leq \max\left(\mathbf{p}^{\mathrm{T}}\begin{bmatrix}\mathbf{P}\\ \underline{t}_j[b]\end{bmatrix}, \mathbf{p}^{\mathrm{T}}\begin{bmatrix}\mathbf{P}\\ \bar{t}_j[b]\end{bmatrix}\right) \\
&\Rightarrow \min_{\mathbf{v}\in\mathcal{R}_i}(\mathbf{p}^{\mathrm{T}}\mathbf{v}) \leq \max_{\mathbf{v}\in\mathcal{R}_j}(\mathbf{p}^{\mathrm{T}}\mathbf{v}),
\end{aligned}$$

with $\mathcal{R}_i, \mathcal{R}_j$ as in (4) for transport assignment i, j, respectively. Similarly, by swapping i and j, we can show that the conditions of the proposition imply that

$$\min_{\mathbf{v}\in\mathcal{R}_j}(\mathbf{p}^{\mathrm{T}}\mathbf{v}) \leq \max_{\mathbf{v}\in\mathcal{R}_i}(\mathbf{p}^{\mathrm{T}}\mathbf{v}).$$

The above two conditions combined imply that $\mathcal{I}_i \cap \mathcal{I}_j \neq \emptyset$. Thus $C = 1 \Rightarrow \mathcal{I}_i \cap \mathcal{I}_j \neq \emptyset$ or equivalently $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset \Rightarrow C = 0$.

Next, we will introduce a binary feature that leads to a required classifier. This feature is based on the orientations of the individual links in a route. It will only be useful if all segments in a route point approximately from start to goal location. Later on, we will address the problem of outliers. Here, we derive a set of required classifiers each based on a binary feature from the orientation. The orientation $\Theta(n_1, n_2) \in [0, 2\pi]$ of an edge $(n_1, n_2) \in \mathcal{E}_{\mathrm{r}}$ is the angle in polar coordinates of the vector $\mathbf{P}(n_2) - \mathbf{P}(n_1)$. We choose a partition of the interval $[0, 2\pi]$. Each element of the partition is related to one binary feature which holds true if the orientation of at least one edge in the route falls in the range of that element. When two routes overlap there must be at least one edge that has the same orientation.

Proposition 4. Let (i,j) refer to the pair of transport assignments. Let $\mathcal{P}$ be a partition of $[0, 2\pi]$. If there is no element $I \in \mathcal{P}$ and edges in the routes of the transport assignments $(n_i[a], n_i[a+1])$, $(n_j[b], n_j[b+1])$ such that $\Theta(n_i[a], n_i[a+1]) \in I$ and $\Theta(n_j[b], n_j[b+1]) \in I$, then $C(i,j) = 0$.

Proof. According to Definition 2, $C(i,j) = 1$ implies that there must be indices a, b such that $\mathbf{P}(n_i[a]) = \mathbf{P}(n_j[b])$ and $\mathbf{P}(n_i[a+1]) = \mathbf{P}(n_j[b+1])$, where n_i, n_j are the node sequences of transport assignment i, j respectively. For these it holds that $\Theta(n_i[a], n_i[a+1]) = \Theta(n_j[b], n_j[b+1])$. Since $\mathcal{P}$ is a partition of $[0, 2\pi]$ and $\Theta(n_i[a], n_i[a+1]) \in [0, 2\pi]$, there must be $I \in \mathcal{P}$ with $\Theta(n_i[a], n_i[a+1]) \in \mathcal{I}$. Since $\Theta(n_j[b], n_j[b+1]) = \Theta(n_i[a], n_i[a+1])$, it follows that also $\Theta(n_j[b], n_j[b+1]) \in \mathcal{I}$. The proof follows from contradiction.

Next, we discuss how we can make this classifier based on orientation more efficient if we can disregard routes that overlap only over a short distance. Apart from the direct reduction in true positives, this approach will also reduce the false-positive rate of the classifiers, since some outlier route edges can be disregarded.

In order to cover the notion that there must be a minimum overlap in routes to be considered, we extend the definition of the coordination function (Definition 2).

Definition 4. (Min. Distance Coordination Function). A coordination function $C : \mathcal{N}_{\mathrm{c}} \times \mathcal{N}_{\mathrm{c}} \to \{0, 1\}$ according to Definition 2 requires minimum distance $l_{\min}$ if the following properties hold: if for a pair (i,j) we have $C(i,j) = 1$, there must be a set of pairs of indices $\mathcal{A}$ such that for all $(a,b) \in \mathcal{A}$ it holds that $\mathbf{P}(n_i[a]) = \mathbf{P}(n_j[b])$ and $\mathbf{P}(n_i[a+1]) = \mathbf{P}(n_j[b+1])$, and $[\underline{t}_i[a], \bar{t}_i[a]] \cap [\underline{t}_j[b], \bar{t}_j[b]] \neq \emptyset$ and $[\underline{t}_i[a+1], \bar{t}_i[a+1]] \cap [\underline{t}_j[b+1], \bar{t}_j[b+1]] \neq \emptyset$. Furthermore we require $\sum_{(a,b)\in\mathcal{A}} \|\mathbf{P}(n_i[a]) - \mathbf{P}(n_i[a+1])\|_2 \geq l_{\min}$.

We adapt the orientation-based classifier (Proposition 4) to exclude links of a total length less than $l_{\min}$. The approach is to calculate the fraction of route length that lies in each element of the partition. We can ignore the intersection with some elements of the partition as long as the lengths of the links whose orientation is contained in these elements sums up to a value less than $l_{\min}/2$.

Proposition 5. Let (i,j) refer to a pair of transport assignments. Let $\mathcal{P}$ be a partition of $[0, 2\pi]$. Let $\mathcal{I}_i \subseteq \mathcal{P}$ and let $\bar{\mathcal{E}}_i \subseteq \mathcal{E}_i$, where $\mathcal{E}_i = \{(n_i[a], n_i[a+1]) : a \in \{1, \ldots, N_i\}\}$, such that for all $e \in \bar{\mathcal{E}}_i$, it holds that there exists $I \in \mathcal{I}_i$ with $\Theta(e) \in I$ and we have $\sum_{(n_1,n_2)\in\mathcal{E}_i\setminus\bar{\mathcal{E}}_i} \|\mathbf{P}(n_1) - \mathbf{P}(n_2)\|_2 < l_{\min}/2$. Similarly, by replacing i by j, we define $\mathcal{I}_j$ for transport assignment j. If $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$, then $C(i,j) = 0$ with C according to Definition 4.

Proof. If $C(i,j) = 1$, then we have a set of pairs of indices $\mathcal{A}$ such that for all $(a,b) \in \mathcal{A}$ it holds that $\mathbf{P}(n_i[a]) = \mathbf{P}(n_j[b])$ and $\mathbf{P}(n_i[a+1]) = \mathbf{P}(n_j[b+1])$. Thus it also holds that $\Theta(n_i[a], n_i[a+1]) = \Theta(n_j[b], n_j[b+1])$. Since $\mathcal{P}$ is a partition of the image of $\Theta(\cdot)$, there is exactly one element $I \in \mathcal{P}$ with $\Theta(n_i[a], n_i[a+1]) \in I$ and since $\Theta(n_i[a], n_i[a+1]) = \Theta(n_j[b], n_j[b+1])$, we have $\Theta(n_i[a], n_i[a+1]) \in I \Leftrightarrow \Theta(n_j[b], n_j[b+1]) \in I$. Furthermore, we have from Definition 4 that $\sum_{(a,b)\in\mathcal{A}} \|\mathbf{P}(n_i[a]) - \mathbf{P}(n_i[a+1])\|_2 \geq l_{\min}$.

Let $\bar{\mathcal{A}}_i$ be a set of the indices of the head nodes of edges in $(\mathcal{E}_i \cap \mathcal{E}_j) \setminus \bar{\mathcal{E}}_i$ paired with the corresponding indices in route j, with $\mathcal{E}_i, \mathcal{E}_j, \bar{\mathcal{E}}_i$ as defined in the proposition. These are the pairs of indices of the edges in the common part of the route that are ignored in transport assignment i. Similarly, let $\bar{\mathcal{A}}_j$ be the index pairs that are excluded due to transport assignment j. We need to show now that $\mathcal{A}$ is not empty without the pairs in $\bar{\mathcal{A}}_i$ and $\bar{\mathcal{A}}_j$, or in other words, that even if the features for either route ignore up to $l_{\min}/2$ of the common part of the route, there are still edges left that let the set of classifiers indicate that the routes intersect. We have from the assumptions made in the proposition $\sum_{(a,b)\in\bar{\mathcal{A}}_i} \|\mathbf{P}(n_i[a]) - \mathbf{P}(n_i[a+1])\|_2 < l_{\min}/2$,

$\sum_{(a,b)\in\bar{\mathcal{A}}_j} \|\mathbf{P}(n_i[a]) - \mathbf{P}(n_i[a+1])\|_2 < l_{\min}/2$, and from Definition 4 that $\sum_{(a,b)\in\mathcal{A}} \|\mathbf{P}(n_i[a]) - \mathbf{P}(n_i[a+1])\|_2 \geq l_{\min}$. Thus $\sum_{(a,b)\in\mathcal{A}\setminus(\bar{\mathcal{A}}_j\cup\bar{\mathcal{A}}_j)} \|\mathbf{P}(n_i[a]) - \mathbf{P}(n_i[a+1])\|_2 > 0$ and since this is a sum over positive elements, we deduce that $\mathcal{A}\setminus(\bar{\mathcal{A}}_j\cup\bar{\mathcal{A}}_j) \neq \emptyset$. But then there is $I \in \mathcal{P}$ and $(a,b) \in \mathcal{A}\setminus(\bar{\mathcal{A}}_j\cup\bar{\mathcal{A}}_j)$ such that $\Theta(n_i[a], n_i[a+1]) = \Theta(n_j[b], n_j[b+1]) \in I$ and thus $\mathcal{I}_i \cap \mathcal{I}_j \neq \emptyset$. By contraposition it follows that $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset \implies C(i,j) = 0$.

It is possible to combine various classifiers as defined in Propositions 3 and 5 in various ways according to Propositions 1 and 2 in Section 3.

5. SIMULATIONS

In this section, we demonstrate in a realistic scenario the method derived in this paper. We demonstrate that the application of 6 classifiers can rule out 99 % of the transport assignment pairs, leaving only 1 % for the computationally expensive exact algorithm. The simulation setup is as follows. The start and goal locations are sampled randomly with probability proportional to an estimate of the population density in the year 2000 (Socioeconomic Data and Application Center (2015)). We limit the area to a large part of Europe. We calculate shortest routes with the Open Source Routing Machine (Luxen and Vetter (2011)). If the route is longer than 400 kilometers, a subsection of 400 kilometers of the route is randomly selected. The maximum speed is $v_{\max} = 80\,\mathrm{km/h}$. We set the start times t^{S} of half the assignments to 0 and sample the start times of the remaining assignments uniformly in an interval of 0 to 24 h. The first half is to account for assignments that are currently on the road while the other half is to account for assignments that are scheduled to depart later. The deadlines t^{D} are set in such a way that the interval $\bar{t}[a] - \underline{t}[a] = 0.5\,h$ where a is any valid index. We consider the minimum length that two assignments have to overlap to be considered for platooning, $l_{\min}$, to be 20 km.

We implemented all features and corresponding classifiers that are described in Section 4, i.e., interval projection (Proposition 3) and minimum distance orientation partition (Proposition 5). Note that Proposition 4 is a special case of Proposition 3 with $l_{\min} = 0$. For interval projection we tested vectors of the form $[1,0,0]^{\mathrm{T}}$, $[0,1,0]^{\mathrm{T}}$, $[0,0,1]^{\mathrm{T}}$, $[1,1,0]^{\mathrm{T}}$, $[-1,1,0]^{\mathrm{T}}$, $\left[-\cos(\alpha), \frac{-\sin(\alpha)}{\cos(0.278\pi)}, \frac{v_{\max}180°}{6371\pi}\right]^{\mathrm{T}}$, with $\alpha = 0, \pi/4, \ldots, 7\pi/4$. The position $\mathbf{P}$ is expressed here as latitude and longitude and measured in degrees. The vectors parametrized by α are approximately orthogonal to a trajectory at maximum speed at the latitude of 50 degrees with heading angle α and should work well for trajectory pairs that have similar orientation, that cover the same area, and that are only separated by a small time margin. We will refer to the corresponding classifiers in the following discussion as $c_{100}, c_{010}, c_{001}, c_{110}, c_{-110}, c_{\alpha 0}, \ldots, c_{\alpha 7}$ respectively. For the orientation-based classifier we use 100 equally sized cells to partition $[0, 2\pi]$. For each cell the fraction of the route distance that falls in this cell is computed. Matches up to $l_{\min}/2$ starting in ascending order of route distance contained in the cells are excluded. We will refer to this classifier as c_{o}.

None	499,500	c_{-110}	108,403	$c_{\alpha 4}$	134,019
c_{100}	104,380	$c_{\alpha 0}$	129,282	$c_{\alpha 5}$	107,287
c_{010}	101,542	$c_{\alpha 1}$	103,240	$c_{\alpha 6}$	105,883
c_{001}	208,896	$c_{\alpha 2}$	103,453	$c_{\alpha 7}$	109,934
c_{110}	98,343	$c_{\alpha 3}$	109,626	c_{o}	453,246

Table 1. Number of positives for different classifiers.

We test $K = 1000$ transport assignments. All classifiers are evaluated in parallel. Then the sequence of classifiers that filters the most assignments at every stage is computed. The number of positives for each classifier is listed in Table 1. Figure 2 shows the number of remaining pairs at each stage, the ground truth, and the sequence of classifiers for this sample. The optimization of the classifier order would typically be done when the system is designed and is to some extent specific for the exact problem setting. In a running platoon coordination system the order in which classifiers are applied would remain fixed.

We can see in Figure 2 that two classifiers, c_{110} and $c_{\alpha 7}$, combined are able to reduce the number of pairs by one order of magnitude. The first classifier, c_{110}, only takes into account longitude and latitude of the routes. The second one, $c_{\alpha 7}$ is orthogonal to the first one, c_{110}, in the plane but also takes into account timing. The third classifier, $c_{\alpha 3}$, is also of the projection type that is able to identify that a pair of assignments cannot platoon if they are geographically close but differ in timing, and it covers the opposite orientation compared to the previous classifier. The fourth classifier, c_{100}, covers a third direction in the plane. It is interesting to see that the fifth classifier, c_0, is the orientation-based classifier. Alone it performs much worse than the other classifiers as can be seen in Table 1. Two transport assignments that take the same route in opposite directions and that "meet" on the way are impossible to identify as a negative with the projection based classifiers. The orientation-based classifier might be able to achieve that. The classifier that only takes into account start and arrival time, c_{001}, is selected last, since most cases it rules out are already covered by the classifiers $c_{\alpha 0}, \ldots, c_{\alpha 7}$ and since half the assignments start at the same time. We see that the benefit from adding more classifiers diminishes quickly as classifiers are added. All classifiers combined can reduce the number of pairs by two orders of magnitude and get within one order of magnitude from the ground truth. The false-positives are mostly very curvy routes that intersect geographically and are separated little in time in the area of the intersection. To be able to correctly identify such pairs as negatives is often not possible with the features presented in this paper. We get fairly consistent results for different runs of the simulation that are omitted here owing to space constraints.

6. CONCLUSIONS AND FUTURE WORK

We presented a method that can significantly reduce the computational effort of centrally coordinating truck platooning over large geographic areas and time intervals. With this framework we can combine different classifiers

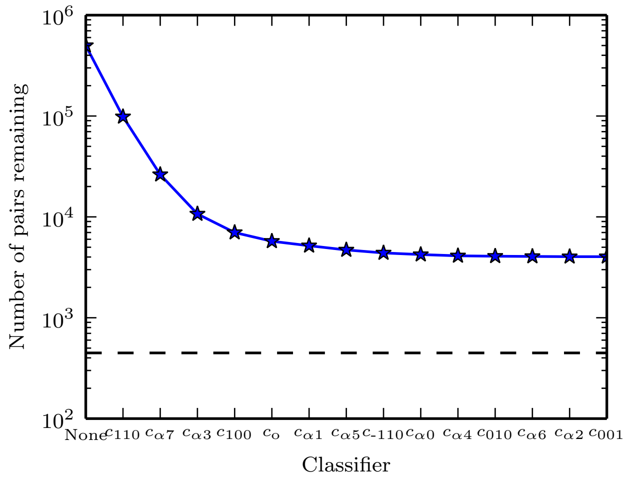

Fig. 2. This plot shows the number of remaining pairs when the classifiers are consecutively applied from left to right. The order the classifiers are chosen in a way that each stage removes as many pairs as possible. The classifier applied at each stage is indicated on the horizontal axis. The dashed line shows the ground truth from the exact algorithm.

to efficiently cull the pairs of transport assignments before they are passed on to an exact algorithm that checks which transport assignments can platoon. We developed three types of classifiers and demonstrated their potential in simulation. This approach might also be useful for dynamically computing the interaction network of spatially distributed multi-agent systems.

There are various directions for future work. While this work focuses on algorithmic efficiency, it would be interesting to tune the implementation for best performance. Furthermore, we would like to gain more insight from simulations and from theoretical analysis on how different parameters affect the effectiveness of the approach. It would be, for instance, desirable to better understand which classifiers should be chosen in which order based on the way transport assignments are generated.

REFERENCES

Agatz, N., Erera, A., Savelsbergh, M., and Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2), 295 – 303.

Asociación Española de Fabricantes de Automóviles y Camiones (2011). European motor vehicle parc. Technical report.

Barthélemy, M. (2011). Spatial networks. *Physics Reports*, 499(1–3), 1 – 101.

Bentley, J. and Ottmann, T. (1979). Algorithms for reporting and counting geometric intersections. *Computers, IEEE Transactions on*, C-28(9), 643–647.

Bonnet, C. and Fritz, H. (2000). Fuel consumption reduction in a platoon: Experimental results with two electronically coupled trucks at close spacing. *SAE Technical Paper*. doi:10.4271/2000-01-3056.

Cohen, J.D., Lin, M.C., Manocha, D., and Ponamgi, M. (1995). I-collide: An interactive and exact collision detection system for large-scale environments. In *I3D '95 Proc. of ACM Interactive 3D Graphics Conference*, 189–196. ACM, New York, NY, USA.

Edelsbrunner, H. and Maurer, H.A. (1981). On the intersection of orthogonal objects. *Inf. Process. Lett.*, 13(4/5), 177–181.

Hall, R. and Chin, C. (2005). Vehicle sorting for platoon formation: Impacts on highway entry and throughput. *Transportation Research Part C: Emerging Technologies*, 13(5–6), 405 – 420.

Hao, H. and Barooah, P. (2013). Stability and robustness of large platoons of vehicles with double-integrator models and nearest neighbor interaction. *International Journal of Robust and Nonlinear Control*, 23(18), 2097–2122.

Horowitz, R. and Varaiya, P. (2000). Control design of an automated highway system. *Proceedings of the IEEE*, 88(7), 913–925.

Jiménez, P., Thomas, F., and Torras, C. (2000). 3d collision detection: A survey. *Computers and Graphics*, 25, 269–285.

Larson, J., Liang, K.Y., and Johansson, K.H. (2015). A distributed framework for coordinated heavy-duty vehicle platooning. *Intelligent Transportation Systems, IEEE Transactions on*, 16(1), 419–429.

Larsson, E., Sennton, G., and Larson, J. (2015). The vehicle platooning problem: Computational complexity and heuristics. *Transportation Research Part C: Emerging Technologies*, 60, 258 – 277.

Liu, E.S. and Theodoropoulos, G.K. (2014). Interest management for distributed virtual environments: A survey. *ACM Comput. Surv.*, 46(4), 51:1–51:42.

Liu, F., Harada, T., Lee, Y., and Kim, Y.J. (2010). Real-time collision culling of a million bodies on graphics processing units. In *ACM SIGGRAPH Asia 2010 Papers*, 154:1–154:8. ACM, New York, NY, USA.

Luxen, D. and Vetter, C. (2011). Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, 513–516. ACM, New York, NY, USA. doi:10.1145/2093973.2094062.

Meisen, P., Seidl, T., and Henning, K. (2008). A data-mining technique for the planning and organization of truck platoons. In B. Jacob, P. Nordengen, A. O'Connor, and M. Bouteldja (eds.), *International Conference on Heavy Vehicles, Heavy Vehicle Transport Technology*, 389–402. John Wiley & Sons, Inc, Hoboken, NJ, USA.

Roy, K. and Tomlin, C. (2006). Enroute airspace control and controller workload analysis using a novel slot-based sector model. In *American Control Conference, 2006*, 6. IEEE, New York, NY, USA.

Socioeconomic Data and Application Center (2015). Population density grid, v3, 2000.

van de Hoef, S., Johansson, K.H., and Dimarogonas, D.V. (2015a). Coordinating truck platooning by clustering pairwise fuel-optimal plans. In *Intelligent Transportation Systems (ITSC), 18th IEEE International Conference on*. IEEE.

van de Hoef, S., Johansson, K.H., and Dimarogonas, D.V. (2015b). Fuel-optimal coordination of truck platooning based on shortest paths. In *American Control Conference (ACC)*, 3740–3745. IEEE, New York, NY, USA.